[COMMITTEE PRINT]

BUGS IN THE PROGRAM

PROBLEMS IN FEDERAL GOVERNMENT COMPUTER SOFTWARE DEVELOPMENT AND REGULATION

STAFF STUDY

BY THE

SUBCOMMITTEE ON
INVESTIGATIONS AND OVERSIGHT

TRANSMITTED TO THE

COMMITTEE ON
SCIENCE, SPACE, AND TECHNOLOGY
U.S. HOUSE OF REPRESENTATIVES

ONE HUNDRED FIRST CONGRESS
FIRST SESSION
Serial G

SEPTEMBER 1989

Printed for the use of the Committee on Science, Space, and Technology

U.S. GOVERNMENT PRINTING OFFICE
21-727 ⇆ WASHINGTON : 1989

For sale by the Superintendent of Documents, Congressional Sales Office
U.S. Government Printing Office, Washington, DC 20402

H702-13

COMMITTEE ON SCIENCE, SPACE, AND TECHNOLOGY

ROBERT A. ROE, New Jersey, *Chairman*

GEORGE E. BROWN, JR., California
JAMES H. SCHEUER, New York
MARILYN LLOYD, Tennessee
DOUG WALGREN, Pennsylvania
DAN GLICKMAN, Kansas
HAROLD L. VOLKMER, Missouri
HOWARD WOLPE, Michigan
BILL NELSON, Florida
RALPH M. HALL, Texas
DAVE McCURDY, Oklahoma
NORMAN Y. MINETA, California
TIM VALENTINE, North Carolina
ROBERT G. TORRICELLI, New Jersey
RICK BOUCHER, Virginia
TERRY L. BRUCE, Illinois
RICHARD H. STALLINGS, Idaho
JAMES A. TRAFICANT, JR., Ohio
LEE H. HAMILTON, Indiana
HENRY J. NOWAK, New York
CARL C. PERKINS, Kentucky
TOM McMILLEN, Maryland
DAVID E. PRICE, North Carolina
DAVID R. NAGLE, Iowa
JIMMY HAYES, Louisiana
DAVID E. SKAGGS, Colorado
JERRY F. COSTELLO, Illinois
HARRY JOHNSTON, Florida
JOHN TANNER, Tennessee
GLEN BROWDER, Alabama

ROBERT S. WALKER, Pennsylvania*
F. JAMES SENSENBRENNER, JR., Wisconsin
CLAUDINE SCHNEIDER, Rhode Island
SHERWOOD L. BOEHLERT, New York
TOM LEWIS, Florida
DON RITTER, Pennsylvania
SID MORRISON, Washington
RON PACKARD, California
ROBERT C. SMITH, New Hampshire
PAUL B. HENRY, Michigan
HARRIS W. FAWELL, Illinois
D. FRENCH SLAUGHTER, JR., Virginia
LAMAR SMITH, Texas
JACK BUECHNER, Missouri
CONSTANCE A. MORELLA, Maryland
CHRISTOPHER SHAYS, Connecticut
DANA ROHRABACHER, California
STEVEN H. SCHIFF, New Mexico
TOM CAMPBELL, California

HAROLD P. HANSON, *Executive Director*
ROBERT C. KETCHAM, *General Counsel*
CAROLYN C. GREENFELD, *Chief Clerk*
DAVID D. CLEMENT, *Republican Chief of Staff*

SUBCOMMITTEE ON INVESTIGATIONS AND OVERSIGHT

ROBERT A. ROE, New Jersey, *Chairman*

JOHN TANNER, Tennessee
JIMMY HAYES, Louisiana
TERRY L. BRUCE, Illinois
HARRY JOHNSTON, Florida

DON RITTER, Pennsylvania
SHERWOOD L. BOEHLERT, New York

GREGORY C. SIMON, *Staff Director and Counsel*
JAMES H. PAUL, *Professional Staff Member*
JOEL B. EISEN, *Republican Counsel*

---

*Ranking Republican Member.

CONTENTS

	Page
Letter of transmittal	V
I. Introduction	1
A. Issues	3
1. Software procurement	3
2. Software assurance	3
3. Maintaining trained personnel	3
4. Professional standards for software developers	3
B. Recommendations	3
II. Software Procurement	4
A. System costs: Hardware versus software	5
B. Software design: Goals and methods	7
C. Software and procurement: Expensive incompatibilities	8
D. Budgeting for design: Adding insult to injury	9
E. The goal: Quality	10
F. Summary	11
III. Software Assurance and Regulation	12
A. Measuring reliability	12
B. Reliability and regulation	13
C. Safety	14
D. Security	16
E. Summary	18
IV. Maintaining trained people and professional standards	19
A. Running the show: The program manager	20
B. Recruitment and retention in public service	21
C. Developing the software professional: Education	23
D. Developing the software professional: Ethics	25
E. Certifying software professionals	26
F. Summary	27
V. Conclusion	28
VI. Recommendations	29
VII. Acknowledgments	32

COMMITTEE ON SCIENCE, SPACE, AND TECHNOLOGY

U.S. HOUSE OF REPRESENTATIVES
WASHINGTON, DC 20515

LETTER OF TRANSMITTAL

July 28, 1989

The Honorable Robert A. Roe
Chairman, Committee on Science, Space,
and Technology
Washington, D.C.

Dear Mr. Chairman:

The staff of the Subcommittee on Investigations and Oversight submits for consideration a report entitled Bugs in the Program: Problems in Federal Government Software Development and Regulation. The report is intended to alert Members of the Committee to the growing problems the Government faces as its dependence on computers increases. Further, computers may pose a threat to public health and safety as their use expands in areas such as medical instrumentation and air traffic control. Agencies charged with regulating these industries have no accepted method for assuring that computer software will operate safely in these applications.

Computers today are caught in a "virtuous cycle." At the same time that each new generation of machines incorporates impressive new capabilities and computing power, the costs of obtaining those benefits is going down. For Government agencies, the purchase of computers is therefore an effective means of raising employee productivity using only modest expenditures. It is this virtuous cycle that underpins the Nation's fundamental defense strategy, where the Department of Defense substitutes highly-capable electronic systems for increased numbers of weapons to maintain the global balance of power.

Unfortunately, the Government does not capture the full benefit of its investment in computer technology because software, the set of instructions necessary to operate the computers, is facing the opposite problem. As the complexity of systems increases, Government managers find that the software they buy or develop does not achieve the capabilities contracted for, that it is not delivered at the time specified, and that the cost is significantly greater than anticipated. Indeed, many recent examples of cost overruns and degraded capability cited as examples of Government waste, fraud and abuse can be related to problems in the development of computer software.

"Government policies," states the report, "on everything from budgeting to intellectual property rights have congealed over time in a manner almost perfectly designed to thwart the development of quality software." The present procurement system frustrates Government program managers by forcing them to write highly-detailed specifications for systems whose boundaries are extremely fluid, by denying them the flexibility to reflect evolving circumstances as the system users gain greater experience (without imposing unacceptable costs on the Government), and by driving them to move to production before all of the vital questions bearing on how the system will ultimately perform have been answered.

If developing software is a problem for Government managers, the problems faced by agencies with regulatory responsibility are even more difficult. To fully understand software and its role in a computer system is difficult unless all of the assumptions underlying the system's structure are clearly understood.

The Federal Aviation Administration has recognized this. In certifying commercial aircraft, agency staff consult with manufacturers throughout the process of design and development to understand why design choices and trade-offs occur. For the Food and Drug Administration, facing hundreds of new computer-driven medical devices, involvement at this level may be impossible. Agency evaluators may never see a software system until it arrives on their desk with a request for approval. At present, the report concludes, there is no infallible method whereby a regulatory agency can assure that the software embedded in a system will not cause death or injury. Developing such a method may be possible, given further research, although no consensus exists or the likelihood of success.

Overcoming these concerns is the responsibility of the Government's software specialists. It is precisely these specialists, however, who are most likely to leave the Government. Low salaries for high--caliber technical experts in government, restrictions on future employment, and the sense that few value the contribution they make to the public welfare are not conducive to retention.

The report also expresses uncertainty regarding the development of future software specialists. It is not clear that present computer science education curricula properly reflect the fact that software development is moving toward team participation as system complexity increases. Also, a lack of concern regarding ethics is a weakness in this profession and contributes to episodes of computer interference such as the recent attack on the Internet research network. It may be in the best interests of the Government to encourage professional certification as part of the strategy for addressing the software "crisis."

No short-term solutions to the problems identified in this report are evident. Recommendations, therefore, proposed by the report reflect a long-range perspective. The primary recommendation involves the creation of an interagency unit in the Office of Science and Technology Policy. This Working Group will identify promising remedies already implemented by agencies and seek to transfer those techniques to other software projects. It will also be responsible for recommending to Congress a research program designed to support the needs of software developers and regulators. Since the software problems identified in this report involve most Federal agencies, are likely to continue for some time, and may represent significant impacts on future appropriations, a White House focus on this area seems warranted.

Secondly, the report recommends revamping NASA's procurement statute to reflect the larger influence of software. This will be coordinated with the consolidated procurement statute now being developed by the Office of Federal Procurement Policy (OFPP) as a consequence of the President's Blue Ribbon Commission on Military Procurement. As one of the agencies most dependent on computers, NASA has much at stake in assuring that agency procedures are set up to build quality in at the outset. NASA's experience should be brought to bear to provide practical guidance to OFPP.

The remaining recommendations involve the missions of other Federal agencies. The National Computer Systems Laboratory of the National Institutes of Standards and Technology is directed to increase work on the application of statistical quality control methods to software development. These techniques have demonstrated their value in manufacturing around the world and some argue that they are equally valid for the production of quality software.

The National Science Foundation, building on its traditional expertise, will continue to provide the basic research that will be needed in improved software. It is interesting to note that Dr. William Wulf, Assistant Director for Computer and Information Science and Engineering at the Foundation, testified before the Senate on July 26, 1989, that "the fundamental intellectual foundation, even the appropriate mathematics, does not exist" to solve the "software crisis."

Finally, the Department of Defense has been funding the Software Engineering Institute at Carnegie-Mellon University as one response to its critical deficiencies in software development. The Institute, however, addresses generic problems affecting all agencies developing or regulating software. The report advocates increased contributions to the Institute from other agencies wrestling with software problems, allowing expansion of SEI's technology transfer efforts. Alternatively, some agencies may believe that it would be better to establish similar capabilities focused on their areas of special concern.

In developing this report, the staff received the full cooperation of many busy people in industry, in universities and in the Federal agencies. Their contributions made this report possible and we express appreciation to all of them.

Respectfully,

JAMES H. PAUL
Professional Staff Member
Subcommittee on Investigations
and Oversight

GREGORY C. SIMON
Staff Director and Counsel,
Subcommittee on Investigations
and Oversight

I. INTRODUCTION

Whenever I see one of these major new programs get behind schedule or go over budget, I know that the programmers just took the gamble and lost.[1]

– MIKE GILBERT, PRESIDENT,
SOFTWARE LEVERAGE

The Federal Government probably buys more computer software than any other single buyer in the United States. Expenditures for software are growing rapidly, as more Federal agencies use computers to meet the responsibilities given them by Congress.[2] As expenditures grow, so do concerns about the reliability, cost and accuracy of ever-larger and more complex software systems.[3] These concerns are heightened as computers perform more critical tasks, where mistakes can cause financial turmoil, accidents or in extreme cases, death.

The potentially tragic consequences of software failure first drew the Investigation and Oversight Subcommittee's attention to issues of software quality and reliability. On April 21, 1986, under the

---

[1] James Kitfield, "Is Software DOD's Achilles' Heel?," *Military Forum*, July 1989, p. 30. Gen. Bernard Randolph, Commander, U.S. Air Force Systems Command, made clear his reason for revamping Air Force software policies when he said, "[W]e have a perfect record on software schedules – we have never made one on time yet and we are always making excuses." "Washington Roundup," *Aviation Week & Space Technology*, February 6, 1989, p. 17.

[2] The Department of Defense "...reach[ed] an annual software expenditure level in mission-critical computer systems [defined as systems whose failure would endanger life, equipment or success of the defined task] of about $9 billion in 1985, with projections of $30 billion annually by 1990." William H. Taft, IV, Deputy Secretary of Defense, Memorandum to the Joint Logistics Commanders, August 12, 1985; cited in *Report of the Defense Science Board Task Force on Military Software*, Office of Under Secretary of Defense for Acquisition, September 1987, p. 6. [Future citations from the Task Force will refer to the *DSB Report.]*

[3] It is instructive to consider how software systems have grown over time. For example, "...software experts point out that the Air Force has gone from the Vietnam War-era F-4, which had virtually no computer code, to the F-16D with 236,000 lines of code." (Kitfield, *loc. cit.)*

Estimates for the original Strategic Defense Initiative (SDI) ranged from 6 million to 100 million lines of computer instructions in its full configuration. No firm analysis has yet defined an actual size, and the program's objectives have significantly changed since its inception on March 23, 1983. The following size estimates are excerpted from various sources:

"The largest single system developed to date for the military was probably Safeguard. It consisted of over 2 million lines of code.... It is estimated that [ballistic missile defense] battle management software would be at least three to five times larger than the Safeguard software." Brockway McMillan, et al., "Battle Management, Communications and Data Processing (U)," *Report of the Study on Eliminating the Threat Posed by Nuclear Ballistic Missiles* (U), Vol. V (Alexandria, Virginia: Defensive Technologies Study Team), October 1983, p. 43.

"It is likely that Phase 1 deployed operational code will total over two million [lines of code].... but I would guess not much more, and its total code, including National Test Bed simulation and software engineering environments, maybe over 25 million, perhaps up to 30 or 40 million." Charles A. Zraket in Ware Myers, "Software Pivotal to Strategic Defense," *IEEE Computer*, January 1989, p. 93.

"[T]he Star Wars battle management system will be by far the most complex body of software ever devised. By one estimate it will require up to 100 million lines of computer code written by hundreds or thousands of individual programmers." M. Mitchell Waldrop, "Resolving the Star Wars Software Dilemma," *Science*, May 9, 1986, p. 710.

The National Aeronautics and Space Administration Space Station program will require a software system capable of developing and supporting several million lines of code – a conservative estimate is that NASA will need five times the current figure of 750,000 lines in space shuttle flight software. See "Space Station Software 'Factory' Planned," *Electronic Engineering Times*, February 29, 1988, p. 4.

chairmanship of Congressman Harold Volkmer, the Subcommittee held hearings to examine the progress being made in medical computer systems[4] and to review attempts by the Food and Drug Administration (FDA) to impose regulation on computer software used in medical devices. These new FDA regulations were required by a 1976 amendment to the Federal Food, Drug and Cosmetic Act of 1938[5] in response to concerns that unsafe or ineffective medical devices might reach the marketplace.

The danger inherent in the growing use of computers in diagnostic and treatment devices was demonstrated by problems with a radiation therapy machine, the "Therac-25," sold by Atomic Energy of Canada, Ltd.. The machine was responsible for exposing patients to massive overdoses of radiation leading to the deaths of at least four people in Texas, Washington and Canada. The source of the problem had been traced to errors in the computer programs controlling the system.[6] The Director of FDA's Office of Science and Technology, Roger Schneider, stated to the Subcommittee staff that the procedures FDA established in response to the Medical Device Amendments of 1976 would not have identified the logical flaw in the Therac software, even if the device had been subject to the most stringent pre-market approval.[7]

FDA is not alone in finding its traditional regulatory efforts lacking in the computer age.

In 1988, the Comptroller General called on the Securities and Exchange Commission (SEC) to improve its ability to carry out "sophisticated technical reviews of the design, development and operation of systems" in the wake of the stock market crash of 1987.[8]

The Nuclear Regulatory Commission (NRC), in 1979, was forced to shut down five nuclear reactors after determining that a software flaw in the computer-aided system used to design the reactors and their cooling systems left the plants vulnerable to seismic damage in earthquakes.[9] The episode led the NRC to include software in the "structures, systems, and components that prevent or mitigate the consequences of postulated accidents that could cause undue risk to the health and safety of the public."[10]

At the present level of understanding in software engineering, Federal agencies cannot be assured the software they oversee and use is correct; they can determine whether the software developer understands good practices that are necessary to produce quality software. Further, review and evaluation of analysis and test results are also useful, though this offers no safety guarantee.[11]

---

[4] *Information Technologies in the Health Care System,* Hearing before the Subcommittee on Investigations and Oversight of the Committee on Science and Technology, U.S. House of Representatives, April 21, 1986; 99th Congress, Second Session, No. 133 (Washington: Government Printing Office, 1986).

[5] Medical Device Amendments of 1976; Public Law 94-295.

[6] Edward J. Joyce, "Malfunction 54," *American Medical News,* October 3, 1986, p. 1.

[7] Briefing at Center for Devices and Radiological Health, Food and Drug Administration, May 19, 1988.

[8] *Financial Markets: Preliminary Observations on the October 1987 Crash,* General Accounting Office, GGD-88-38, January 1988, p. 98.

[9] Nancy G. Leveson, "Software Safety: What, Why and How," *Computing Surveys,* Vol. 18(2), June 1988, p. 127.

[10] 10 CFR 50, Appendix B.

[11] Telephone conversation with Ken Meyer, Systems and Equipment Branch, Northwest Intermountain Regional Office, Federal Aviation Administration, September 17, 1987.

A. Issues

Several issues face the Congress in the area of computer software. These include problems in software procurement, improving software reliability, maintaining trained personnel in government software programs and developing professional standards for those designing and developing software. This report, the result of a study by the Investigations and Oversight Subcommittee staff, examines these issues.

1. Software Procurement

Getting value for the Government's money is a recurring problem. Computer software, which is now a major cost item in many procurements, is not immune from traditional procurement problems such as delay, cost overruns and poor performance. Worse, the procurement system as presently structured does not take into account the special needs of computer software systems and can compromise effective software development from the start.

This is critical, since software development is a design-intensive activity and the cost of correcting errors increases markedly as development proceeds. This report discusses how the present procurement system may contribute to software reliability problems.

2. Software Assurance

Federal regulation of software is the second topic highlighted in this report. As computers take on more critical functions, issues of safety and security have arisen. This report examines this debate, and discusses the impact that software research will have on Federal regulation of software.

3. Maintaining Trained Personnel

The successful development and management of software depends on the people involved. As software complexity increases, Federal agencies will find themselves in need of well-trained software specialists. They will compete with the private sector for these people. Finding enough people who can do the job in a booming market is a daunting prospect, especially for regulatory agencies.

4. Professional Standards for Software Developers

This report closes by examining the status of professional development in the software field and whether the Nation's computer science and software engineering education programs are incorporating the necessary topics and techniques into their curricula. It will also discuss action by the software community to increase professionalism among its members.

B. Recommendations

Seeking solutions to the problems outlined in this report, five recommendations are made. First, the Office of Science and Technology Policy (OSTP) should oversee a program of long-term interagency coordination and cooperation in software. Many Federal

agencies have already worked through the pitfalls of increasing complexity in software systems, but other agencies are not benefiting from that experience. Technology transfer, in the experience of the Committee on Science, Space, and Technology, requires direct contact between technical professionals. This recommendation seeks to create a forum where this can occur.

The second major change involves the system of government procurement. It is a theme of this report that software cannot be properly developed using the welter of regulations presently in force. While software now drives system requirements, the procurement system still focuses attention on hardware. NASA and the Office of Federal Procurement Policy are asked to design new software-oriented procurement regulations reflecting today's reality in buying systems.

Other recommendations involve the missions of specialized agencies in the Government and their contributions to solving software problems. The National Computer Systems Laboratory of the National Institutes of Standards and Technology should take the lead in adapting statistical quality control to software development. For NSF, the report reaffirms its mission to perform basic research and improve education. Finally, the programs of the Software Engineering Institute (SEI) should be expanded beyond a focus on defense needs, as they address problems common to most Federal agencies. Some agencies facing extensive involvement in software may find it useful (or necessary) to duplicate certain SEI capabilities in their own organization.

II. SOFTWARE PROCUREMENT

> There is no new thing under the sun.
>
> – ECCLESIASTES 1:9

Procurement reform, especially in the Department of Defense, is a perennial topic of Congressional concern. As noted by the Committee on Armed Services, "Four decades have produced six different commissions with six reports on the acquisition process.... [T]hese repeated investigations have come up with similar proposals for reforming the system."[12] The Defense Department is not alone in getting less than it paid for: examples can be found at the Internal Revenue Service, the Veterans Administration, the Social Security Administration; the list is endless. Today, what has become apparent is that system costs are driven not by hardware, but by software.

Norman Augustine, Chairman and Chief Executive Officer of the Martin Marietta Corporation, highlighted this in his humorous study of Government and industry procurement foibles, *Augustine's Laws*. Focusing on the development of tactical military aircraft,

---

[12] *Defense Acquisition: Major U.S. Commission Reports* (1949-1988), Volume 1, Report Prepared for the Use of the Defense Policy Panel and Acquisition Policy Panel of the Committee on Armed Services, U.S. House of Representatives; 100th Congress, Second Session, Committee Print 26 (Washington: Government Printing Office, 1988), p. vii.

where costs are exploding[13] and electronic systems are increasingly used to obtain the highest level of performance,[14] he concludes:

> Optimally, what is needed is something that can be added to airplanes and other systems which weighs *nothing*, is *very costly*, yet violates none of the physical laws of the universe.... This might appear to be an insurmountable challenge. However, as a result of the traditional ingenuity characteristic of American engineers it can be reported with confidence that such an ingredient has already been found.... *software.*[15]

"Software," Augustine says, "is like entropy. It is difficult to grasp, weighs nothing and obeys the Second Law of Thermodynamics; i.e., it always increases."[16]

Why does the Government's software cost so much? Software development is a complex process that requires modern "systems engineering" techniques. Further, this process is a special problem for the system of statutes and regulations that form the framework of the Government's acquisition methods. Government policies on everything from budgeting to intellectual property rights have congealed over time in a manner almost perfectly designed to thwart the development of quality software. Highlighted here are major contributors to software disasters: the method for bidding on Government contracts and the impetus to minimize up-front costs in program budget estimates.

A. System Costs: Hardware vs. Software

"The procurement establishment is undoubtedly more comfortable dealing with hardware than software," stated the editors of *Aviation Week & Space Technology* in October, 1988.[17] And yet, hardware is diminishing in importance when budgeting for system procurement. The following is a graphical summary of data on trends in hardware and software costs:

---

[13] "In the year 2054, the entire defense budget will purchase just one aircraft. This aircraft will have to be shared by the Air Force and Navy 3-1/2 days each week except for leap year, when it will be made available to the Marines for the extra day." Norman R. Augustine, *Augustine's Laws* (New York: Viking Penguin Inc., 1986), p. 111.

[14] "After the year 2015, there will be no airplane crashes. There will be no takeoffs either, because electronics will occupy 100 percent of every airplane's weight." *Ibid.*, p. 96.

[15] *Ibid.*, p. 116.

[16] *Ibid.*, p. 118.

[17] "Achilles Heel," *Aviation Week & Space Technology*, October 17, 1988, p. 15.

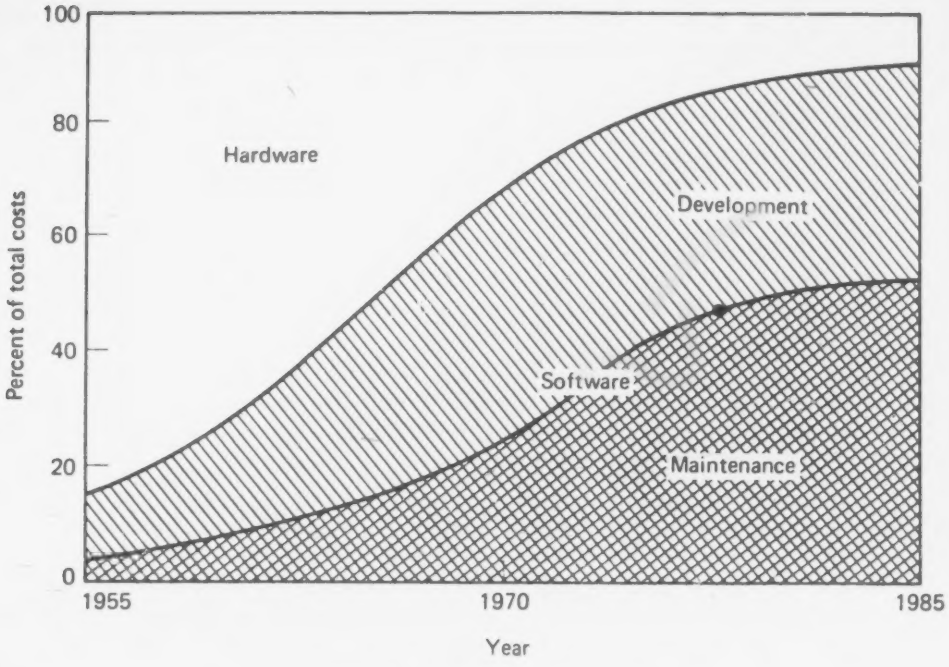

Figure 1: Hardware/Software Cost Trends[18]

A prime example of this orientation can be seen in the Strategic Defense Initiative. In the form originally proposed, it would be the largest and most complex software development project devised by the Federal Government. From the outset of the program, it was recognized that

> *The battle management system* will, *through its software, define and control the functioning of the entire defense and,* thereby, *define its effectiveness and establish performance requirements for weapons and sensors....The battle management system and its software must be designed as an integral part of the BMD system as a whole, not as an applique.* [emphasis added][19]

However, only four years after initiation of the program, the Defense Science Board concluded,

> *The Strategic Defense Initiative... has a monumental software problem* that must be solved to attain the goals of the initiative. It is critical quite out of proportion to its cost.... *Initial contractor proposals... largely ignored it.... No program to address the software problem is evident.* [emphasis added][20]

SDI progress, though, is usually discussed in terms of hardware. In May, 1986, Dr. Gerold Yonas, Chief Scientist for the Strategic Defense Initiative Organization, spoke of

> ... [I]mpressive demonstrations of a missile intercepting another missile in the outermost atmosphere,... substantial progress in passive, active and interactive discrimination.... One experiment – in which a laser was used against a model booster – yielded important information about the extent to which boosters could withstand a laser attack.[21]

---

[18] Barry W. Boehm, *Software Engineering Economics* (Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1981), p. 18. Reprinted with permission.
[19] McMillian, *et. al., op. cit.*, p. 4.
[20] *DSB Report*, p. 23.
[21] Gerold Yonas, "SDI: A Progress Report," *Center* Magazine, May/June 1987, p. 8.

The widening gap between hardware and software performance exacerbates the problem of cost.[22] For the software designer, improved hardware creates a need for longer, more complex programs and prevents software costs from dropping as rapidly as hardware costs.[23] *Project and procurement managers* who focus on hardware first and consider software only as an afterthought soon learn that "software is now the choke point in large systems."[24]

B. Software Design: Goals and Methods

Software is a design-intensive activity. The development process begins with an attempt to define the functions the software will be expected to control or execute as part of a system. Clarity and stability in system goals are imperative to assure quality in the software. The Strategic Defense Initiative is also an example of how changes in hardware design or missions can lead to extensive changes in software. When the Government commences highly technical programs on a grand scale it should be expected that any programmatic changes will lead to major redesign of software undergoing parallel development.

Some argue that hasty design is the main cause of software error,[25] others that the problem comes from attempting to specify in detail a system that no one knows how to build.[26]

Both views are part of the consensus among software professionals that the most effective method for reducing problems in software is to allow sufficient time for design. Fred Brooks says,

> *The hardest single part of building a software system is deciding precisely what to build....* No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.
>
> *Therefore, the most important function that the software builder performs for the client is the iterative extraction and refinement of the product requirements.* For the truth is, the client does not know what he wants. The client usually doesn't know what questions must be answered, and he has almost never thought of the problem in the detail necessary for specification.... *So in planning any software design activity, it is necessary to allow for an extensive iteration between the client and the designer as part of the system definition.* [emphasis added][27]

This consensus has led to changes in the models describing the software development process. Classically, software engineers employed the so-called "waterfall model,"[28] which assumed that the system was fully defined before software design began. Newer models, like Boehm's "spiral model,"[29] advocate repeated testing of

---

[22] Otis Port, et al., "The Software Trap: Automate or Else," *Business Week*, May 9, 1988, p. 142.

[23] Frederick P. Brooks, Jr., "No Silver Bullet," *IEEE Computer*, April 1987, p. 10.

[24] Fred Reed, "STARS Director: Hardware Specs Stymie Software," *Federal Computer Week*, November 28, 1988, p. 37.

[25] "Almost every error in software can be attributed to a specification activity. *A design that is hastily put together has many negative results, not the least of which is an inordinate amount of software change requests* (SCRs). These are indicators of errors in the design phase and a portent of things to come." Nick P. Stewart, Software Quality Engineering Supervisor, Satellite and Space Division, Rockwell International Corporation, "Software Error Costs," *Quality Progress*, November 1988, p. 48. [emphasis added]

[26] Telephone conversation with Col. Joseph Greene, USAF, Director of the Software Technology for Reliable, Adaptable Systems (STARS) program, Department of Defense, July 21, 1988.

[27] Brooks, *op. cit.*, p. 17.

[28] The diagram representing the process was usually drawn as a cascade of steps from design to implementation, hence the name.

[29] Barry W. Boehm, "A Spiral Model of Software Development and Enhancement," *IEEE Computer*, May 1988, p. 61.

various alternative software designs to determine which best addresses user needs within resource constraints and at acceptable risk. Upon completion of this process, the final design will then proceed through full-scale development. The so-called "evolutionary development" model advocates delivery of intermediate designs to the end user. This approach permits developers to "adjust both design and objectives based on observed realities."[30]

These models reflect the commonsensical principle that any process can be improved with practice. Unfortunately for the software specialists at agencies like NASA, this new view of software development conflicts with the realities of the present procurement system.

C. Software and Procurement: Expensive Incompatibilities

The Defense Science Board Task Force on Military Software recently pointed out the supreme irony in software procurement: software is purchased for the flexibility it offers, yet the Government acquires software using a system whose hallmark is rigidity. The Task Force comments:

> The most common present method of formulating specifications -- issuing a Request for Proposal, accepting bids, and then letting a contract for software delivery -- is not in keeping with good modern practice and accounts for much of the mismatch between user needs and delivered function, cost and schedule....
>
> *[W]e now understand the importance of iterative development of requirements, the testing of requirements against real users' needs by rapid prototyping, and the construction of systems by incremental development, with early incremental releases subjected to operational use....* [emphasis added][31]

The Task Force recommended changes in the old standard in an attempt to "...remove any remaining dependence upon the assumptions of the "waterfall" model and to institutionalize rapid prototyping and incremental development...."[32] In response to the Task Force, the Department of Defense, on February 29, 1988, adopted the revised "Defense System Software Development" Military Standard, better known in the software community as DOD-STD-2167A. The standard is expected to become the Government's primary guidance for contracts requiring software development. Still, the Task Force noted that

> Although some parts of the recent Draft DOD-STD-2167A appear to encourage modern methods, *the draft as a whole continues to reinforce exactly the document-driven, specify-then-build approach that we believe causes so many of DoD's software problems.* [emphasis added][33]

Out on the technical frontiers, then, the Government requires a legally-binding contract specifying in excruciating detail exactly how the system will look at delivery some years hence. To maintain schedules, the software subcontractor begins writing the system's requirements. Inevitably, changes are introduced. Accordingly, new requirements are generated. This adversely affects software quality and increases development costs.

---

[30] Tom Gilb, *Principles of Software Engineering Management* (Wokingham, England: Addison-Wesley Publishing Company, 1988), p. 84.
[31] *DSB Report*, p. 33.
[32] *Ibid.*, pp. 33-34.
[33] *Ibid.*, p. 33.

Ms. Jane Radatz,[34] Program Manager for Development of DOD-STD-2167A, stated at a recent conference that "DOD-STD-2167A provides... [f]lexibility to allow many design methodologies."[35] Despite her efforts to convince industry representatives that the new standard does incorporate the change in thinking about software design, profound skepticism evidently still exists. Whether the new, improved standard will be interpreted by Government managers in a manner reflecting this need for flexibility can be answered only by experience.

D. Budgeting for Design: Adding Insult to Injury

The lack of flexibility in system design is now compounded by the pressure a program faces from the budget process. Program costs are often underestimated at the outset "because of the pressure to get the money flowing" in a tight budget.[36] This hurts the program at the crucial design and development stage and further complicates long term planning.[37] As Stewart states, "It is estimated that it can cost six to ten times as much to correct an error found during the maintenance phase[38] as it would have in the design phase. In some highly complex aerospace applications, the estimates go as high as 100 times."[39]

This clearly indicates that "[w]e must increase our tolerance for failure, but be able to find those failures early, when they are cheap. *This can be done by conducting critical experiments on systems options prior to making the final acquisition decision.*"[40]

There is here a fundamental dissonance. In order to reduce the total lifetime cost of a system, more time -- and money -- must be spent asking questions in the design stage. Experiments should be run between competing alternatives. Prototype software systems should be developed and tested with actual users to determine whether the systems requirements reflect actual needs. Getting approval for the program, on the other hand, means running a gauntlet of agency managers and Members of Congress demanding that first-year costs be minimized.

> [T]he government is searching for the best performance a given design can offer for a given number of development dollars. However, *because of the severe competition for the few large programs, there is a significant tendency for firms to "buy in" -- that is, bid low....* They believe that *if their promised performance satisfies the government and if their development bid is the lowest, they will win the development contract -- and they usually do.* [emphasis added][41]

---

[34] Engineering Applications Department Manager, Strategic and Information Systems Division, Logicon, Inc.

[35] Jane W. Radatz, "Software Standards Review and Discussion, Session 1: DOD-STD-2167A Software Development Standard (SDS)," *Tutorial Proceedings: National Security Industrial Association Fall Joint Conference on Software Quality*, October 18, 1988, p. 316.

[36] Rodney D. Stewart and Ann L. Stewart, *Managing Millions: An Inside Look at High-Tech Government Spending* (New York: John Wiley and Sons, Inc., 1988), p. 61.

[37] "Because there is always a limit on the amount of money Congress will appropriate, [an agency], in an effort to begin as many programs as possible, often sets optimistically low cost estimates for proposed programs.... [T]he underestimation of initial costs exacerbates the difficulties associated with long-term planning." J. Ronald Fox, *The Defense Management Challenge: Weapons Acquisition* (Boston: Harvard Business School Press, 1988), pp. 64-65.

[38] The maintenance phase is that period after the software has been released and is in use in the field.

[39] Nick Stewart, *loc. cit.*

[40] James E. Colvard, "Getting Technical: How to Buy Weapons the Smart Way," *The Wall Street Journal*, January 31, 1989, p. A23. [Emphasis added]

[41] Jacques Gansler, *The Defense Industry* (Cambridge, Massachusetts: The MIT Press, 1982), p. 74.

The results of these policies can be seen in the Air Force B-1B strategic bomber. Flight tests show that the aircraft's defensive electronic countermeasures system does not achieve the capabilities established by the Air Force in the B-1B contract. There are "basic problems with the system's architecture.... [S]ystem integration and software development for the ALQ-161 [radar receiver processor] [were] unable to keep pace with concurrent development and production...."[42] The House Armed Services Committee recited a familiar litany of problems in its oversight report:

> In announcing its two-bomber program... in October 1981, the Administration agreed to limit the total B-1B program cost (in fiscal year 1981 dollars) to $20.5 billion.... *To meet the cost cap, the aircraft had to be produced at the most economical production rate,* four per month. In allowing that rate to "drive" the program, the Air Force was required to accept a high degree of concurrency. *Development, production, and testing of the B-1B occurred in a highly compressed schedule. Decisions on the production of critical B-1B systems occurred before development and testing had advanced to the point where systems problems could be found and fixed.* [emphasis added][43]

The Comptroller General reported in 1986 that the Federal Aviation Administration's (FAA) Advanced Automation System -- the foundation of the nation's ability to manage future air traffic -- was being developed with an approach that

> *does not adequately mitigate technical risks and does not provide for suitable operational simulation of the advanced automation features....* [S]ignificant uncertainty exists about the Advanced Automation System's cost growth. The program has already encountered significant cost growth (at least 50 percent in the design competition phase) and... *FAA's acquisition strategy increases the risk of further cost growth.* [emphasis added][44]

E. The Goal: Quality

The Defense Science Board Task Force spoke for all Government managers when it reported that the software problem was "that we cannot get enough of it, soon enough, reliable enough, and cheap enough."[45]

A proposed solution, not universally accepted, would apply to software development the same statistical quality control methods advocated by W. Edwards Deming and used with remarkable success in Japanese industry.[46] One software development process called "clean room" is now attempting to employ statistical methods to improve the reliability of software. This method is intended to break down the design process into increments, test each increment statistically, and generate data that can prove that the process is producing the expected product.[47]

---

[42] "Tests of B-1B ECM System Uncover Basic Design Flaws," *Aviation Week & Space Technology*, July 18, 1988, p. 31.

[43] *The B-1B: A Program Review*, Report of the Panel on the B-1B, Subcommittee on Research and Development and Subcommittee on Procurement and Military Nuclear Systems, Committee on Armed Services, U.S. House of Representatives, 100th Congress, First Session; Committee Print No. 2 (Washington: Government Printing Office, 1987), p. 11.

[44] *Air Traffic Control: FAA's Advanced Automation System Acquisition Strategy is Risky*, General Accounting Office, IMTEC-86-24, July 1986, p. 3.

[45] *DSB Report*, p. 6. "Users often complain that the software they receive from [management information systems] takes too long to build, costs too much, does less than expected (and is too expensive to enhance), and has too many errors that are too time consuming to have fixed." Richard Zultner, "The Deming Approach to Software Quality Engineering, *Quality Progress*, November 1988, p. 58.

[46] Zultner, *loc. cit.* See also Harlan D. Mills and J. H. Poore, "Bringing Software Under Statistical Quality Control", *Quality Progress*, November 1988, p. 52.

[47] Mills, *loc. cit.*

This "clean room" method may offer a second benefit: it would permit managers to accommodate changes in user requirements, described earlier as one of the main causes of flawed software. Constant focus on user needs is maintained.

A statistical approach is used for testing which is more natural from the user standpoint than the developer's. Test cases which reflect statistical samples of user operations provide observable times between failures of the software that are relatable to the operating environment. On the other hand, *testing carried out by selected test cases, no matter how carefully and well-planned, can provide nothing but anecdotes.* From experience, user-oriented testing has been shown to provide a much better indication of what can be expected with product use than would be included by developer selected test cases. [emphasis added][48]

Given the prevailing view that poor communication between user and developer explains most software flaws, methods that promote user involvement force both sides to resolve misunderstandings at the earliest, and cheapest, opportunity for corrections.

F. Summary

The Government's present system for procuring software does not meet the Government's needs and wastes resources. The application of "systems engineering" disciplines is needed to remedy the procurement system's defects.

The President's Blue Ribbon Commission on Defense Management advocated that "Federal laws governing procurement should be recodified into a single, greatly simplified statute applicable government-wide." The Office of Federal Procurement Policy (OFPP) has been attempting to implement this recommendation, according to GAO.[49] If this new, simplified statute is to successfully incorporate the current thinking on software development techniques, it "may well look quite foreign to someone trying to understand [it] from the mindset of today's normal software engineering and systems acquisition process."[50]

Critical to the new procurement statute must be a "system-first" perspective. It is important that consideration for system requirements drive managers. Allowing hardware and software development to proceed separately or in isolation is a formula for problems. Giving software equal status in planning for procurement will certainly change Government procurement. Good systems engineering, where the program manager factors in user needs, safety and security at the outset of design and seeks tradeoffs to match his available resources, may leave nothing tangible to show when the time comes for the next budget cycle. No program manager relishes the thought of defending a request for funds when the major activity seems to be endless arguments over abstruse technical points by large numbers of well-paid engineers. Yet experience shows this is precisely the course to follow because it answers most, if not all, the questions that are expensive to fix on a production line.

---

[48] P. Allen Currit, Michael Dyer and Harlan D. Mills, "Certifying the Reliability of Software", *IEEE Transactions on Software Engineering*, January 1986, p. 4.

[49] *Defense Management: Status of Recommendations by Blue Ribbon Commission on Defense Management*, General Accounting Office, NSIAD-89-19FS, November 1988, p. 38.

[50] STARS Joint Program Office, "SOW Materials Attachment A5: Software-First Systems Development Standard for Systems-in-the-Large (Draft)," Software Technology for Advanced, Reliable Systems Request for Proposal No. F19628-88 R-0011, October 29, 1987, p. 4.

III. SOFTWARE ASSURANCE AND REGULATION

The genesis of this report, as described in the introduction, was the tragic circumstances created by a software error in the computer control system of a radiation therapy machine. For the Food and Drug Administration, responsible for assuring the safety and effectiveness of medical devices, the Therac-25 is a harbinger of problems that will only grow in the future. FDA is not alone, of course. Other agencies are using computers in areas that bear on the health and welfare of Americans and are struggling to adapt their procedures to address the questions posed by software.

Ideally, an agency would like to apply a standard analysis to a software-driven system and receive an unambiguous evaluation of its potential for creating hazards. Such tools do not now exist. Agencies essentially rely on "...codes of good practice. They give no assurance of reliability."[51] The current effort to bring mathematical precision to what now is more art than science will benefit the regulatory agencies by making it possible to identify failure-prone software before it is released.

Statistical quality control, a set of techniques which has proven its worth in the manufacturing arena, is now being applied in software development. Implicit in the idea of statistical quality control, as Kelvin said, is the ability to measure.

> When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind: it may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced to the stage of *science*.[52]

Today's software development requires "increasingly sophisticated methods of measurement and analysis."[53]

A. Measuring Reliability

Keeping the Nation's telephone system operating is a difficult and demanding task. The advent of computers improved the ability to keep the system working even as the volume of calls multiplied. It is therefore not surprising that AT&T has produced pioneering work in software reliability.

Reliability in software can be defined as "the probability of failure-free operation of a computer program for a specified time," where a failure represents "a departure of a computer program's operation from the user's requirements."[54] AT&T's major contribution was the relation of reliability to the number of failures that occur in a software program during the time it is being executed by a computer's processor unit. Execution time is more closely related to the chance that software will fail than normal time.[55]

---

[51] Peter Mellor, "Can you Count on Computers?," *New Scientist*, February 11, 1989, p. 55.

[52] Emily Morison Beck, Ed., *Bartlett's Familiar Quotations*, 15th Edition (Boston: Little, Brown and Co., 1988), p. 594.

[53] David N. Card, "Major Obstacles Hinder Successful Measurement," *IEEE Software*, November 1988, p. 82.

[54] John D. Musa, "Tools for Measuring Software Reliability," *IEEE Spectrum*, February 1989, p. 39.

[55] John D. Musa, Anthony Iannino, and Kazuhira Okumoto, *Software Reliability: Measurement, Prediction, Application* (New York: McGraw-Hill Book Company, 1987), p. 248.

The AT&T model has shown promising results. A 1986 in-house project demonstrated the ability to predict, with an error of one week, the date on which the software would achieve an acceptable level of failures during use. The Hewlett-Packard Company, which is working with AT&T to expand research in reliability models, applied the model to software embedded in two new computer terminals, and found that the results were within 12 percent of predicted values.[56]

B. Reliability and Regulation

For a Federal agency concerned about the regulation of software, reliability is a basic requirement. Software that exhibits abnormally high numbers of failures, even where such failures cause only inconvenience, cannot be trusted to perform critical control or monitoring functions. Actually, the problem facing a regulatory agency resembles the problem facing an agency buying software – what is the capability of the organization supplying the software and does the delivered product meet the defined requirements?

A method for addressing these issues has been developed by the Software Engineering Institute established by the Department of Defense as part of the Consolidated Software Initiative.[57] DOD's new software development standard[58] continues the requirement for assessing the software engineering capability of a bidding organization. SEI has defined a set of five "maturity levels," ranging from an "initial process level" where an organization has barely advanced beyond simple programming and debugging, to the "optimizing process level" reflecting an organization that understands software engineering, applies support technology effectively and analyzes statistical data to identify areas for improvement.[59]

A regulatory agency applying this model could learn much useful information. These assessment techniques could be applied by a regulatory agency seeking to determine whether software was developed using good engineering practices:[60]

> [T]he practice of software engineering is not only complex but is still evolving and is not yet fully defined....
>
> The process activities and data [collected in an assessment]... are used as indicators of software engineering capability.... *It is not intended that either the process activities or data be identified as deliverable items in a procurement contract....*
>
> [T]he state of the practice is measurable and... this state can be compared to a norm. [Assessment] will be used initially to establish the norm. [emphasis added][61]

---

[56] Musa, "Tools for Measuring...," p. 39.

[57] In addition to the SEI, the Consolidated Software Initiative encompasses the Software Technology for Adaptable, Reliable Systems program referred to earlier in this report, and the Ada Joint Program Office, organized to manage the development and evolution of the new computer language established as a DOD standard. (Ada is a registered trademark of the U.S. Department of Defense, Ada Joint Program Office.) The Initiative attempts to respond to the growing software woes experienced by the Department.

[58] DOD-STD-2167A.

[59] Suzanna Garreau, "The SEI Builds a Framework for Software Development Process Management," Software Engineering Institute *Bridge*, Vol. 2(3), Fall/Winter 1987, pp. 1, 2.

[60] Briefing by Watts Humphrey, Director, Software Process Program, Software Engineering Institute, Carnegie-Mellon University, August 16, 1988.

[61] W.S. Humphrey and W.L. Sweet, "A Method for Assessing the Software Engineering Capability of Contractors," Technical Report CMU/SEI-87-TR-23, Software Engineering Institute, September 1987, pp. 10-11.

Establishing that norm requires progress in the development of measures like those under study in software reliability. Unfortunately for regulators, the software community still has far to go:

> [R]eliability measurements are not, at present, part of the formal software development methodologies used by NASA or the U.S. Department of Defense. And *theorists caution that such mission-critical applications demand levels of reliability too high for existing models.* [emphasis added][62]

Assignment of actual numerical values to FAA software failure probabilities, for example, is a source of sharp debate in the software community.[63] According to the Radio Technical Commission for Aeronautics:

> During the preparation of this document [DO-178A], techniques for estimating the post-verification probabilities of software errors were examined. The objective was to develop numerical requirements for such probabilities for digital computer-based equipment and systems certification. *The conclusion reached,* however, *was that currently available methods do not yield results in which confidence can be placed to the level required for this purpose.* [emphasis added][64]

On the other hand, agencies cannot afford to ignore Voltaire's warning that the best is the enemy of the good. While problems exist with quantitative assessment, this argues for more attention to assessment, not less. Until quantitative assessment techniques are improved, qualitative assessment can provide important insight into a software development organization and can indicate the quality of a software product. That is better than the present situation.

C. Safety

Safety is the primary concern of a regulatory agency. System designers have seized on the virtues of software, especially its flexibility, in order to achieve better performance. These systems, however, are increasingly used for controlling equipment or processes where failure has the potential to destroy human life. Growing attention has been devoted to helping regulators catch up with the development of such critical software. Dr. Nancy Leveson, of the University of California at Irvine, was first to define the problem of software safety:

---

[62] Musa, *op. cit.*, p. 42.

[63] The FAA defined event probabilities in "System Design Analysis," Advisory Circular 25.1309-1, Federal Aviation Administration, September 7, 1982, as follows:

PROBABLE: Probable events may be expected to occur several times during the operational life of each airplane. A probability on the order of 1 in 100,000 or greater.

IMPROBABLE: Improbable events are not expected to occur during the total operational life of a random single airplane of a particular type, but may occur during the total operational life of all airplanes of a particular type. A probability on the order of 1 in 100,000 or less.

EXTREMELY IMPROBABLE: Extremely improbable events are so unlikely that they need not be considered to ever occur, unless engineering judgment would require their consideration. A probability on the order of 1 in one billion or less.

Many of those attending the 1988 Computer Assurance conference challenged the FAA's ability to measure a one-in-one-billion chance of occurring when identifying "extremely improbable" events during the certification process. Most in the audience applauded the suggestion to eliminate quantitative definitions. See Jonathan Jacky, "Compass '88 Trip Report," Draft, July 18, 1988, p. 5.

[64] Special Committee 152, *Software Considerations in Airborne Systems and Equipment Certification,* Radio Technical Commission for Aeronautics, Document DO-178A, March 1985, p. 6. [Future citations will reference RTCA/DO-178A.]

The goal of software safety is the avoidance of system safety failures[65] which are caused by a software error and/or are detected and handled by software procedures. A *safe system* is one which prevents unsafe states from producing safety failures where an *unsafe state* is defined as a state which may lead to a safety failure unless some specific action is taken to avoid it. [emphasis in original][66]

Reliability is a necessary but insufficient condition to assure software safety. Leveson cautions that assigning numbers for reliability cannot be the end of safety analysis for regulators. She elaborates:

Reliability is usually defined as the probability that a system will perform its intended function for a specified period of time under a set of specified environmental conditions.... [S]afety is the probability that conditions that can lead to a mishap *(hazards)* do not occur, whether or not the intended function is performed.... In general, reliability requirements are concerned with making a system *failure free*, whereas safety requirements are concerned with making it *mishap free*. These are not synonymous.... *Not all software errors cause safety problems, and not all software that functions according to specification is safe.... Severe mishaps have occurred while something was operating exactly as intended, that is, without failure."* [emphasis added on first and final sentences][67]

It is the thrust of the software safety argument that issues of safety must be raised in the context of the full system, and that the principles of systems engineering apply equally to software and hardware. Michael Brown, Chairman of the Department of Defense Triservice Software Systems Safety Working Group, describes it as "[t]he optimization of Systems Safety in the design, development, use and maintenance of software systems and their integration with safety critical hardware systems in an operational environment."[68]

What may appear to be trivial design decisions early in a program's lifetime may later have serious consequences. The tragedy of Iran Air Flight 655, shot down by missiles from the USS *Vincennes* on July 3, 1988, illustrates this. The investigation report notes "the importance of the information being presented by way of the USS VINCENNES Large Screen Displays (LSD) became apparent.... [I]t is important to note that altitude cannot be displayed on the LSD in real-time."[69] Analysis of stored data showed that the Iranian aircraft climbed continuously from takeoff to missile impact. Had the Aegis system displayed altitude data on the target display, the captain might have seen that the aircraft was not descending toward his ship despite the reports he was receiving from the anti-aircraft warfare station. Ships carrying the Aegis system are now being refitted to correct this deficiency.[70]

The Triservice Systems Safety Working Group has proposed a program plan attempting to improve the Government's ability to

---

[65] Defined by Leveson as "...a failure which leads to casualties or serious consequences. The definition of 'serious consequences' will have to be left up to the system designer."

[66] Nancy G. Leveson and Peter R. Harvey, "Analyzing Software Safety," *IEEE Transactions on Software Engineering*, September 1983, p. 570.

[67] Nancy G. Leveson, "Software Safety: What, Why and How," *Computing Surveys*, Vol. 18(2), June 1986, pp. 135-136.

[68] Michael L. Brown, "Software Systems Safety and Human Errors," *Proceedings of the Computer Assurance Conference: COMPASS '88*, Institute of Electrical and Electronic Engineers, Inc., p. 20.

[69] William M. Fogarty, RADM, USN, *Investigation Report: Formal Investigation into the Circumstances Surrounding the Downing of Iran Air Flight 655 on 3 July 1988,"* U.S. Department of Defense, July 28, 1988, p. 3.

[70] Lynn Elber, "Rogers Ends Command Without Apology," The Associated Press, May 27, 1989.

manage the risks posed by software. A research and development center would be established to fund research into software analysis and risk assessment techniques. An Applications Laboratory would be responsible for transferring the tools and methods developed by the research center into practice.[71] The proposal has not been funded by the Department of Defense, but its value for regulatory agencies attempting to assure that software in a regulated environment will be safe argues that a similar organization should be created by the Federal Government.

D. Security

Computer system security is a growing concern, highlighted in November 1988 by the partial collapse of Internet, the international computer network. A small program placed on the network, allegedly by a graduate student at Cornell University, proceeded to duplicate itself at unexpected speed. The resulting mass of message traffic ultimately overwhelmed much of the system.[72] The incident was ultimately harmless, although aggravating to the systems managers around the country who struggled to purge the invader from their systems.

What might happen? In California, Kevin Mitnick has been convicted of software theft by telephone from a Santa Cruz, California, company, is suspected of retaliating against a firm that denied him a job by sending false information to a financial news computer service[73] and has now pleaded guilty to electronic theft of a computer program from the Digital Equipment Corporation and illegal use of MCI long-distance access codes.[74] The Lawrence Berkeley Laboratory, while attempting to locate the source of a series of computer break-ins, found that "[the intruder]... had entered a computer used in the real-time control of a medical experiment. Had we not caught him in time, a patient might have been severely injured."[75] In 1983, a group of students in Milwaukee succeeded in gaining access to radiation-therapy records stored on computers at the Memorial Sloan-Kettering Cancer Center in New York. No data in these treatment profiles was changed and no patients were injured.[76]

In 1985, the Bank of New York ended one trading session in Government securities overdrawn by $23.6 billion. The problem was a software fault that led the computer system to stop transferring securities to other banks; to balance its books, the bank had to borrow from the Federal Reserve Bank of New York. The interest on the loan came to $5 million.[77] The reaction of the Government

---

[71] Triservice Systems Safety Working Group, "Proposal for Software Systems Safety Research and Development," January 13, 1988, p. 10.

[72] John Markoff, "'Virus' in Military Computers Disrupts Systems Nationwide," *New York Times*, November 4, 1988, p. A1.

[73] "Drop The Phone," *Time*, January 9, 1988, p. 49.

[74] Linda Deutsch, "Government Strikes Plea Bargain With 'Dangerous' Hacker," The Associated Press, March 15, 1989. Judge Mariana Pfaelzer of the U.S. District Court in California rejected the plea bargain and ordered Mitnick to trial, saying that Mitnick had "...been engaged in this activity for far too long and no one has adequately punished you." Linda Deutsch, "Judge Rejects Plea Bargain as Too Lenient," The Associated Press, April 25, 1989.

[75] Clifford Stoll, "Stalking the Wily Hacker," *Communications of the ACM*, vol. 31(5), May 1988, p. 489.

[76] William D. Marbach, "Beware: Hackers at Play," *Newsweek*, September 5, 1983, p. 42.

[77] Bob Davis, "Costly Bugs: As Complexity Rises, Tiny Flaws in Software Pose a Growing Threat," *The Wall Street Journal*, January 28, 1987, p. 25.

securities market was, according to E. Gerald Corrigan, then President of the New York Federal Reserve Bank, "a backup in the willingness and ability of some other market participants to transfer securities among themselves."[78] Regulators might consider the chaos that would ensue should a determined attacker bring down the Government securities network and the New York Stock Exchange trading network simultaneously.

Reacting to previous incidents of unauthorized access to Government information systems, the Science, Space, and Technology Committee developed and Congress passed the Computer Security Act of 1987.[79] The Act required Federal agencies to identify their computer systems containing "sensitive" information and directed the National Institute of Standards and Technology (NIST) and the National Security Agency (NSA) to provide advice and comment after reviewing the proposed security plans for those systems.

Responding to a request from this Committee, the General Accounting Office reported in May 1988 that, having reviewed nine system development projects,

> GAO observed common elements among the problems at the agencies studied. *Many of these elements could be traced to the fact that in the early stages of development no agency treated information security as one of the systems' integral functional requirements in a manner similar to other user requirements.* [emphasis added][80]

One commentator argues that

> *[T]he solution is to integrate Accreditation* [the process of security activities required for system acceptance] *into the Acquisition Life Cycle and assure that all security requirements are specified as a functional baseline for subsequent design, development and test.* This solution forces security to be collectively treated top-down as any other set of requirements, to be addressed in each life cycle phase, baseline, review, audit and test as part of the Acquisition Life Cycle process. [emphasis added][81]

Boehm's "spiral" software development model, modified to include the security attributes of confidentiality, integrity and availability, offers one method for achieving that goal.[82] Using that model, designers would assure that security attributes are included in the information architecture analysis,[83] that a risk analysis incorporated security policies or directives, parameters and architecture[84] and that security requirements, as defined in the system design, were considered during actual development phases.[85]

Though networks have been the focus of major concern in computer security, regulators cannot afford to overlook possible intrusions into stand-alone systems. Computer control systems in nuclear power plants, for instance, will not normally be networked. Yet corrupting such a system could lead to an accident with the release of radiation into the environment. On a more ominous note, "at

---

[78] *Ibid.*
[79] Public Law 100-235.
[80] *Information Systems: Agencies Overlook Security Controls During Development*, General Accounting Office, IMTEC-88-11, May 1988, p. 3.
[81] William Norvell, "Integration of Security Into the Acquisition Life Cycle," Presentation before the National Security Industrial Association Fall Joint Conference on Software Quality, October 20, 1988, p. 6.
[82] Harold J. Podell, "Integrating Security Into Software Engineering for Information Systems in Development," January 1989 (unpublished monograph).
[83] *Ibid.*, pp. 8-9.
[84] *Ibid.*, p. 10.
[85] *Ibid.*, pp. 11-12.

one point it was discovered that some parameter in the code for the Minuteman ICBM was discovered to be set to 'the worst possible value.' This was felt to be 'statistically unlikely.'"[86] Standards developed under the Computer Security Act should be studied quite diligently by regulators, because they will be needed. As Dr. Leveson notes, "[s]afety and security are closely related."[87]

E. Summary

Regulation is a difficult task; software regulation even more so. The lack of readily available quantitative methods and techniques leave agencies no choice but to rely on qualitative oversight. This is not comfortable for someone who knows that human life may depend on the system being regulated. Success in the development of a quantitative method demonstrating correctness of software would be welcome to a number of agencies.

Safety and security are system-wide properties and concerns. Their importance to regulators indicates that, like procurement project managers, regulators will find it necessary to involve themselves early in the system design process. FDA has issued 64 recall notices for computerized medical devices through July 1987, mostly because "something was incorporated into the software during the development, some defect.... The defects weren't found until the device was distributed."[88] An agency relying on system testing to find software failures is waiting until far too late in the process to act. The cost of correcting errors at those stages is sure to be met with industry complaints about burdensome regulation.

The FAA requires a system hazard analysis at the outset of the aircraft certification process, seeking to identify the most critical failures that might occur and the causes of those failures.[89] The aircraft design can then be modified to remove the cause or to provide for mitigating the effects. The analysis is particularly vital to FAA, allowing it to concentrate its limited resources on the most severe risks, especially those supporting essential aircraft functions.[90] According to Leveson, however, the process includes no explicit consideration of software, unlike DOD.[91] FAA staff told Dr. Leveson they were contemplating adding similar tasks to their

---

[86] Jacky, *op. cit.*, p. 6.

[87] Leveson, *op. cit.*, p. 137.

[88] *GMP Inspection of Software*, Transcript of the Joint Meeting of NEMA's Division of Diagnostic Imaging and Therapy Systems and the United States Food and Drug Administration, Washington, D.C., July 14, 1987, p. 11.

[89] Briefing by Ken Meyer, Northwest Intermountain Regional Office, Federal Aviation Administration, November 13, 1987.

[90] An "essential" function, as defined in FAA Advisory Circular 25.1309-1, is one "whose failures would contribute to or cause a failure condition which would significantly impact the safety of the airplane or the ability of the flight crew to cope with adverse operating conditions." They are more difficult to analyze than "non-essential" functions, posing no threat, or "critical" functions, where the threat is self-evident. Essential functions may actually be unrecognized critical functions, depending on the circumstances that arise when a failure occurs in systems supporting that function.

[91] Telephone conversation with Dr. Leveson, July 28, 1989. Task 202 of DOD MIL-STD-882B, Notice 1 (Preliminary Hazard Analysis), states that "contractors shall perform a preliminary hazard analysis... during the concept exploration phase or earliest life cycle phases.... *This shall include consideration of the potential contribution by software (including software developed by other contractors) to subsystem/system mishaps.*" [emphasis added] U.S. Air Force Systems Command, "Military Standard: System Safety Program Requirements," MIL-STD-882B Notice 1, March 30, 1984, p. 202-1.

process. In any event, this process is a logical model for other regulatory agencies to emulate.

A final, more philosophical concern regarding regulation was raised by Professor Jonathan Jacky of the University of Washington at the 1988 Computer Assurance Conference. FDA, he points out, deals with a number of very small firms whose entire product line may consist of one device and whose corporate roster may have no more than ten people. The software development methods described throughout this report, on the other hand, are derived from aerospace and defense firms with thousands of technical specialists. Can these techniques be scaled down?[92] M. Frank Houston, in an article entitled "What Do the Simple Folk Do," tries to demonstrate that the systems engineering techniques applicable to software in the very large can also work at the level of the very small.[93] To assure safety, will it be necessary to raise the cost of entry to the market? It is a question that deserves serious consideration.

IV. MAINTAINING TRAINED PEOPLE AND PROFESSIONAL STANDARDS

Behind every successful software system, be it for air traffic control or medical diagnosis, is a software manager. Like the software itself, these managers must be developed. The government's difficulty in finding trained, talented program managers has created a "quiet crisis" plaguing the Civil Service.

The failure of the Government to offer starting salaries for computer and software specialists that are competitive with those in the private sector may be leading to a situation in which the Government will be unable to attract the "best and brightest" members of this field. At the same time, the ability of the Government to attract senior executives into Government service is reduced by recent ethical reform sharply limiting an individual's ability to return to the private sector after a period of public employment.[94] The Government must compete for a limited pool of expertise in specialties like software engineering and is badly handicapped in doing so. Critics have warned that failure to maintain a technically-trained Government workforce will ultimately damage the public interest.

If the exemplar of the software professional is no longer a programmer operating in isolation, what does this portend for educat-

---

[92] Jacky, *op. cit.*, p. 7.

[93] M. Frank Houston, "What Do the Simple Folk Do?," *Supplement to the Proceedings of the 1987 Computer Assurance Conference*, Institute of Electrical and Electronic Engineers (Washington Section), June 15, 1987, p. S-20.

[94] At NASA, for example, "[i]nternal studies show that 45% of NASA's top 2,500 senior and middle managers are eligible for retirement immediately. Within two years the percentage will grow to nearly 70%." (Craig Covault, "NASA Faces Personnel Crisis, Long-Term Funding Shortage," *Aviation Week & Space Technology*, February 20, 1989, p. 23.) Recently, the agency suffered resignations among its corps of senior managers, including the Space Station *Freedom* program manager, his deputy, and the agency's chief scientist. (Doug Isbell, "Senior NASA Executives Resign Over Procurement Integrity Rule," *Washington Technology*, May 11, 1989, p. 27.)

Other agencies are not immune: The Secretary of Energy was turned down twice by candidates for the post of assistant secretary for defense programs, while the Department of Defense has interviewed at least twenty-four candidates for the post of under secretary for acquisition. Judith Havemann, "U.S. Pay Levels Blamed as Key Jobs Go Unfilled," *Washington Post*, May 17, 1989, p. A4.

ing new generations of computer scientists and software engineers? Indications exist that not much has changed in this field of late, meaning that many poor practices are still being taught. Progress, however, is occurring.

A. Running the Show: The Program Manager

"As the key figures in the acquisition process," says Fox, "program managers must oversee the efforts of their [agencies] to acquire, deploy or operate, and support major... systems of proven capability within approved schedules and budgets."[95] Competence in this position is critical since

> ...*[P]rogram success may hinge on the extent to which requirements are flexibly stated and subject to feedback from the design and development phases.* The program manager and contracting officer should have sufficient latitude over the... requirement[s] to
>
> *establish a creative design process in industry,*
> *ensure a competition among design alternatives,* and
> *permit trade-offs in performance and cost as new information becomes available.* [emphasis added][96]

Reality for the typical program manager has little correspondence to the Comptroller General's ideal. Fox, interviewing a DOD program manager, demonstrates a number of differences:

> *The program manager has very little flexibility.* He is given a program and resources. *He does not have the flexibility to trade off among the parameters of technical performance, cost and schedule.* He can recommend changes. But he has very limited authority other than that to grant an exemption from some minor specifications and standards.... In DOD *there is nobody really in charge* as is a contractor's program manager. Instead, *we have many interest groups that can influence the system.* There are so many checks and balances that decisions are very slow. Many can say no and very, very few can say yes. [emphasis added][97]

Colvard notes that Government acquisition teams today are being led by managers specializing in administration or finance, not the technical areas involved in the program. "Neither the present generation of acquisition managers or their staffs," he says, "have up-to-date technical expertise."[98] At DOD, program managers are selected from military officers whose career path has reached a stage where they are ready for management. Whether the officer involved has technical expertise applicable to the program he is assigned to manage is a secondary consideration. A Navy captain could find himself leaving command of a destroyer and managing the next-generation antiradar missile development contract with only a short break to attend the Navy program manager's development school.[99]

In other cases, contract recommendations made by the technical staff are overridden for reasons having more to do with politics or budget. Amtrak, for example, has been attempting to develop an automated switching system to permit increased traffic on the heavily-traveled Northeast Corridor between Washington and Boston. Software has been a difficult problem for the contractor,

---

[95] Fox, *op. cit.*, p. 153.
[96] *DOD Acquisition: Strengthening Capabilities of Key Personnel in Systems Acquisition*, General Accounting Office, NSIAD-86-45, May 1986, p. 41.
[97] Fox, *op. cit.*, p. 156.
[98] Colvard, *loc. cit.*
[99] Jane Radatz, personal conversation, October 18, 1988.

Chrysler. Chrysler is not typically the first company that comes to mind when discussing software questions. According to James Early, Amtrak's project engineer, they were also not the recommended choice for the contract. They were chosen, however, after Amtrak was directed by the Federal Railroad Administration to accept the bid. The reason: Chrysler came in with the lowest price.[100]

Program managers are also expected to "tailor" the myriad standards and requirements imposed by Government contracts based on their knowledge of technical risk and other factors. Fully imposing a complex standard like DOD-STD-2167A on every piece of software purchased by the Government would drive costs through the roof, would produce a mass of useless paper and would reduce safety in critical systems by diverting management attention. Yet in some cases, Government Requests For Proposals are issued asking *contractors* to recommend how the standards should be applied to the contract.[101] Such contracts may reflect a lack of sufficient attention to system design by the Government and should be monitored by agency managers to avoid serious trouble in the program.

B. Recruitment and Retention in Public Service

At the same time the Government is watching its software needs explode, it finds that it cannot attract the people it needs to assure that it receives quality products. The Defense Science Board Task Force noted that "DoD's demand for software capability grows exponentially. It does so at a greater rate than the combined growth in the size and productivity of the pool of software personnel."[102]

At NASA, the Safety, Reliability, Maintainability and Quality Assurance (SRM&QA) Headquarters Office has only two people to try to oversee software assurance activities in programs spanning all of NASA's field centers. This was scheduled to be reduced to one on December 31, 1988. The agency decided to retain the second position.[103] Briefings at the Johnson Space Center (JSC) in October 1987 stated that their SRM&QA office had recently been increased to 18 Civil Service employees, supported by 24 to 36 contract staff supplied by the Boeing company. The JSC staff stated that SRM&QA had previously lacked skills needed for proper oversight of software development activities, but that they were taking steps to overcome that handicap.[104] The Federal Aviation Administration has hired two industry specialists, one in avionics and one in software, as National Resource Specialists to assist in the certification of aircraft and equipment used in American airspace.[105]

---

[100]Telephone conversation with James Early, Chief Engineer, Northeast Corridor Traffic Control System, Philadelphia, Pennsylvania, January 27, 1989.

[101]Meeting of Working Group on DOD Handbooks, National Security Industrial Association Fall National Joint Conference on Software Quality, San Diego, California, October 19, 1988.

[102]*DSB Report*, p. 38.

[103]Telephone conversation with Donald Sova, Information Systems Software Assurance Program Manager, National Aeronautics and Space Administration, December 19, 1988.

[104]Briefing by Dennis Horn, Safety, Reliability, Maintainability and Quality Assurance, Johnson Space Center, October 5, 1987.

[105]Briefing at the Northwest Intermountain Regional Office, Federal Aviation Administration, November 13, 1987.

The Comptroller General recently summarized the reasons why Government agencies cannot find the technically-skilled people necessary to produce quality software.[106] It is no surprise that the major finding is that Government salaries are far below the prevailing salaries in industry. GAO cites

> [a] Merit Systems Protection Board's recent survey [that] *found the government's entry-level salaries for computer specialists ranged from 22.7 to 31.6 percent less than those in private industry.*[107]

The disparity becomes worse; for experienced senior management the Government is 65 percent behind the private sector.[108] William Wilson of NASA pointed out that engineers can be hired at the GS-9 salary schedule, whereas computer scientists enter at a GS-7 level. He went on to say that NASA sometimes could only recruit people because they were fascinated by the space program and were willing to sacrifice the higher level of pay available in the private sector.[109] Linda Ragsdale[110] reports that, so far, NASA has coped by finding methods to classify incoming software specialists as engineers, thereby qualifying them for special pay levels authorized by Congress. Computer scientists not eligible for the special pay levels number 82 out of NASA's workforce of 22,000, and represent those whom the agency cannot find ways to upgrade to engineer. She believes that the quality of incoming recruits is still adequate for NASA's needs.[111]

This strategy may lose its effectiveness if the current rhetoric on the status and competence of the career Civil Service continues. The Comptroller General states that denigration of Government employees contributes significantly to turnover among the technically-skilled people the Government must retain.[112] Too many lucrative and intellectually stimulating opportunities exist for the Government's best technical and managerial people if they believe their service is not appreciated by the public.

Stemming the "brain drain" in an era when "the shortage of trained personnel is already apparent and is projected to become severe by the year 2000"[113] will take the following steps, concludes the Comptroller General. Senior Government managers must:

> *Develop,* as a high priority, *a strategy to hire and train senior-level, technically oriented managers, and develop career paths that will provide technically trained executives* who can successfully introduce state-of-the-art technology....
>
> *Develop strategies for recruiting and keeping technically skilled, junior-level staff.*
>
> *Establish special salary scales for technical personnel.* [emphasis added][114]

The level of technical expertise, in general, available to the Government is a continuing concern of the Committee on Science,

---

[106] *Transition Series: Information Technology Issues,* General Accounting Office, OCG-89-6TR, November 1988, pp. 18-21.
[107] *Ibid.,* p. 19.
[108] *Ibid.*
[109] Interview with William Wilson, Manager, Software Management and Assurance Program, National Aeronautics and Space Administration, September 24, 1987.
[110] Supervisor for Staffing and Classification Policy, National Aeronautics and Space Administration.
[111] Telephone conversation with Linda Ragsdale, July 28, 1989.
[112] GAO, *Information Technology Issues,* pp. 19-20.
[113] *Ibid.,* p. 19.
[114] *Ibid.,* pp. 20-21.

Space, and Technology.[115] Committee action on the problem includes new personnel policies at the National Institute of Standards and Technology and a similar search for innovative hiring and salary structures at NASA. Further efforts along these lines will be needed in other technically oriented mission and regulatory agencies.

C. Developing the Software Professional: Education

"One clear sociological trend," writes Taz Daughtrey,[116] is the professionalization of the software enterprise.... A necessary, but insufficient, condition for accepted professionalism is adherence to official standards of practice."[117] Education is the seedbed for good practice. Whether computer science education in American universities is contributing to or inhibiting good software practice is not clear. Where present needs call for software specialists capable of working in interdisciplinary fields, training today is delivered on an individual basis and in a manner promoting lack of attention to design in favor of immediate attempts to generate code.

Software engineering education has not changed much of late, at least to one practitioner. The field still resembles a "shoot-from-the-hip industry" locked in an academic debate about its status as a discipline separate from computer science. Most students are "not taught an engineering environment, just coding."[118]

Encouragingly, these concerns are not being ignored. Examples of action can be found in Federal agencies, professional societies, and at universities. The National Science Foundation established the Directorate for Computer and Information Science and Engineering, reflecting the increased importance of the field and giving it stature equivalent to other NSF programs.[119] The Foundation also convened a workshop at the George Washington University in March 1988 to solicit assistance from the community in improving undergraduate computer science education.

That panel, in a report endorsed by the IEEE Computer Society Board of Governors,[120] concluded, "Undergraduate computer science education in the United States has problems -- serious ones -- which must be fixed.... [C]omputer science research is already underfunded, both in absolute terms and in comparison to other scientific disciplines."[121] They recommended Federal support for the development and dissemination of new teaching technologies and curricula, funding assistance for improving laboratory equipment, and the establishment of teacher awards, like those given to Presi-

---

[115]See, for example, the statement regarding the technical capability of NASA's space shuttle management structure in *Investigation of the Challenger Accident*, Report of the Committee on Science and Technology, U.S. House of Representatives, 99th Congress, 2nd Session: House Report 99-1016 (Washington: Government Printing Office, 1986), pp. 154-156.

[116]Lead Analyst, Nuclear Power Division, Babcock and Wilcox.

[117]Taz Daughtrey, "The Search For Software Quality," *Quality Progress*, November 1988, p. 29.

[118]Telephone conversation with Nick Stewart, Chairman, Software Division, American Society for Quality Control, July 7, 1988.

[119] David Gries, Terry Walker and Paul Young, "1988 Snowbird Report: A Discipline Matures," *IEEE Computer*, February 1989, p. 73.

[120]"Society's Board Backs Report Urging Upgrading Computer Science Schooling," *IEEE Computer*, September 1988, p. 75.

[121]James Foley, ed., "Undergraduate Computer Science Education: Report of a Workshop Sponsored by the National Science Foundation," March 10-11, 1988, p. 20.

dential Young Investigators, to draw attention to the field.[122] Workshop participants believed that one problem in obtaining these increased resources is the perception that Congress "had given huge sums to computer science education in setting up the four NSF supercomputer centers.... Like many people, they associated the funding of computers with the funding of computer science and engineering."[123] However, the Committee on Science, Space, and Technology has sought to double funding for the National Science Foundation and has achieved significant increases in the Foundation's budget. The Directorate for Computer and Information Science and Engineering budget has benefited from increased resources. If computer science education is not receiving sufficient support, this may be a problem with the mechanism for allocating directorate resources at NSF.

New curricula are being developed. The Department of Computer Science at the University of Tennessee has developed a curriculum based on the "clean room" disciplines propounded by Harlan Mills. Dr. Jesse Poore, department director and Mills's collaborator, explained that students operate as project teams and are evaluated based on the success of the first software execution run.[124] The Software Engineering Institute, described earlier in this report, is chartered to "...influence software engineering curricula development throughout the education community."[125] According to Norman Gibbs, Director of the Education Program, the goals of the SEI effort are to "increase the number of highly qualified software engineers... [and] be the leading center of expertise for software engineering education and training."[126] SEI will accomplish this mission by recognizing that:

> The audience for software engineering education is large and diverse.... [and] Each audience needs a different curriculum. Therefore, *we made an early decision to define the content of software engineering education in modules. Each module presents a relatively small and highly focused topic....* The Education Program has a small permanent staff, so we have relied on visiting educators and software engineers to develop the curriculum modules. [emphasis added][127]

Gibbs believes his program will succeed because it does most of the preliminary work needed to develop an effective teaching plan and because SEI will be supplying high-quality supporting materials.

Whether software engineering should stand apart from computer science is still an open question. The Association For Computing Machinery (ACM), in a task force report on "the core of computer science," argued that it should not. Software methodology and engineering was one of nine "subareas"[128] identified as "vital enough

---

[122]*Ibid.*, pp. 7, 12, 15, 18.

[123]Gries, *et. al., op. cit.*, p. 74.

[124]Mills cites examples from IBM, the Air Force and NASA to support his assertion that "[h]uman verification can produce software robust enough to go to system test without debugging." Mills and Poore, *op. cit.*, p. 55.

[125]Norman R. Gibbs, Briefing at the Software Engineering Institute, Presentation Chart Aug1EDneg1, August 16, 1988.

[126]Gibbs, Presentation Chart Aug1EDneg2.

[127]Gary Ford, Norman Gibbs and James Tomayko, "Software Engineering Education: An Interim Report from the Software Engineering Institute," Software Engineering Institute Technical Report CMU/SEI-87-TR-8, May 1987, p. 2.

[128]Defined as a segment of computer science with a unity of subject matter and incorporating elements of theory, abstraction and design. Peter J. Denning, et. al., "Computing as a Discipline:

Continued

to create and maintain a body of literature...."[129] Programming, they stated, should remain part of the computer science curriculum. Given the symbiotic relationship between computer hardware and software, a case can be made that those in the field should be competent in both. On the other hand, there may be some benefit to a more focused software engineering curriculum, especially in supplying critical skills now woefully short in the Federal Government. A closer examination of this seems warranted.

D. Developing the Software Professional: Ethics

Recent incidents point up a second area bearing on the question of professionalism. Part of the mantle of a professional derives from adherence to a code of ethical behavior defined by the community. A muted debate over ethics in the software field began with the 1983 Sloan-Kettering incident discussed earlier,[130] but is growing in intensity following the November Internet incident.

There is a distinct vacuum in the treatment of ethics in computer science. Jay "Buck" BloomBecker, Director of the National Center for Computer Crime Data, writes of a student who said "'being ethical only allows other people to take advantage of you.'" BloomBecker continued:

> I shouldn't have been surprised. The student who spoke was *enrolled in a computer science program that offered no ethics courses.* As such he was like virtually every other computer student in the U.S. A mere handful of schools offer computer ethics courses.
>
> *Faculty members praise me for coming to town to speak about... computer ethics. They explain that with all the important changes in technology, they just don't have the time to teach that subject.* [emphasis added][131]

The President of the IEEE Computer Society and Dean of the School of Engineering at Vanderbilt University, Dr. Edward A. Parrish, issued a statement in the wake of the Internet attack that stated, "[t]he vast majority of professional computer scientists and engineers, who design, develop, and use these sophisticated networks, are dismayed by this total disregard of ethical practice and forfeiture of professional integrity."[132] At the other extreme, Herwart "Wau" Holland, founder and leader of the Chaos Computer Club of Hamburg, West Germany, stated that "We believe we have the right of access to information, and we take it."[133] James Ball, Stanford University's director of computing, best explained the most likely state of affairs, saying "the underlying ethical and legal questions about why spreading such a virus was wrong, or even whether it was wrong, haven't been discussed much, reflecting a [*sic*] important weakness in the discipline."[134]

Final Report of the ACM Task Force on the Core of Computer Science, in Cooperation with the IEEE Computer Society," *IEEE Computer,* February 1989, p. 65.

[129]*Ibid.*

[130]Vivian Aplin-Brownlee, "Ethical Questions Arise From Computers Biting Into Privacy," *Washington Post,* May 23, 1984, p. A1.

[131]Jay BloomBecker, "Computer Ethics for Cynics," *Computerworld,* February 29, 1988, p. 17.

[132]Edward A. Parrish, "Statement on Responsibility of Computing Professionals," IEEE Computer Society, November 25, 1988, p. 2.

[133]Gail Schares, "A German Hacker's Club That Promotes Creative Chaos," *Business Week,* August 1, 1988, p. 71.

[134]Rick Gladstone, "Computer Virus Raises Ethical Questions at Leading Schools," The Associated Press, November 8, 1988.

It is apparent that "...the [computing] field is not educating its younger people about their professional responsibilities the way other people do."[135] Cornell University officials conducting the inquiry into the Internet incident found that

> By any reasonable standards, the acts involved in the creation and distribution of [the Internet] worm were selfish and wrong and violated the trust that exists between members of the computer science community....[136]

Mr. Fletcher Buckley of GE Aerospace argues instead[137] for greater emphasis on the concepts and activities developed as part of the National Security Agency's "Trusted Security" effort.[138] He argues "[t]his will make a much bigger dent in the 'intrusion by hackers' problem than professional ethics courses."[139] There should be no objection to complementary use of both better security planning and a more explicit focus on professional ethics.

E. Certifying Software Professionals

Entwined with issues of education and ethics is a growing debate among the professional societies about instituting professional certification as a prerequisite for membership. This is a sensitive subject. Even the hint that the growing criticality of software applications should lead the Government to license software engineers provoked indignant reactions. "I'll fight them to the death," said Robert Ulrikson, president of Logical Services, Inc., of Santa Clara, California.[140] Dr. Leveson states she might support certification in the specific case of safety-critical projects, as is now the case in Great Britain.[141] Dr. Joyce Little, vice chairman of the Institute for the Certification of Computer Professionals, said

> *People are going into programming with no background....* All you have to do is hang out your shingle. It's pretty horrifying.... *[I]f the industry does not police itself by means of voluntary certification, then there is going to be licensing. I am not necessarily for licensing.* I see it as a necessary evil that is going to come. [emphasis added][142]

Resolving the debate is of interest, since it is the right of the Government, when contracting for the purchase of software, to include any provisions that a prudent customer believes will assure a quality product. If certification or licensing will achieve that goal, then it is in the best interests of the Government to require that contractor personnel have these credentials. The community is also realizing that it faces the possibility of state regulation that is

---

[135]Gries, *et. al., loc. cit.*

[136]Ted Eisenberg, *et al.,* "The Computer Worm: A Report to the Provost of Cornell University on an Investigation Conducted by the Commission of Preliminary Enquiry," Cornell University, February 6, 1989, p. 41.

[137]Fletcher Buckley, comments prepared July 21, 1989, page 2.

[138]Department of Defense, "Department of Defense Trusted Computer System Evaluation Criteria," DOD 5200.28-STD, December 1985.

[139]Buckley, *loc. cit.*

[140]Pete Carey, "Programmers May Be Facing Regulation," *San Jose Mercury-News,* December 26, 1987, p. 1A.

[141]Telephone conversation with Dr. Leveson, July 28, 1989. The Ministry of Defence in the United Kingdom has adopted MoD-Std-0055, which "...will require that an engineer sign off on the software's safety compliance, that the engineer have taken accredited formal-methods instruction within the past two years, and that an independent engineer with similar accreditation also sign off on the system." Galen Gruman, "Software safety focus of new British standard," *IEEE Software,* May 1989, p. 95.

[142]Carey, *op. cit.,* p. 20A.

...haphazard, inconsistent and conflicting.... It is time for ACM, with IEEE and others if possible, to propose and strive for adoption of appropriate federal standards for the computing profession.[143]

The Subcommittee on Investigations and Oversight recently conducted hearings on integrity in scientific research,[144] with a principal focus on the quality of scientific self-governance. This Committee believes that a professional community should be permitted to enforce its own standards so long as it demonstrates fair, impartial and expeditious consideration of these questions. Failure of the software community to accept their responsibility in this area may lead to the loss of their prized autonomy.

F. Summary

Rejection of the pay raise proposed by the Quadrennial Commission on Executive, Legislative and Judicial Salaries returned the debate on the Civil Service to the *status quo ante.*[145] The National Commission on the Public Service has issued a report with recommendations for improving the pay, flexibility and status of the technical professionals at the heart of Federal agencies.[146] The Committee's experience in trying to implement changes in personnel systems in agencies under its jurisdiction does not offer much solace to those who hope for speedy implementation of the Commission recommendations. The Federal deficit poses an especially difficult barrier to pay increases, the most direct stimulus to agency morale. But if the Government is not able or unwilling to pay for expertise, it will lose that which it already has and will certainly be unable to find substitutes. Actively driving away the people capable of managing the complex systems which underpin human safety and economic soundness in modern America is ludicrous to contemplate, but it may be the *de facto* legacy of the Carter and Reagan Administrations.

Concern for the image of the software professional, in the wake of recent attacks by computer hackers, has sparked debate on improving standards of practice in the field. Arguments have been rekindled on the value of certification in assuring competence. While this is a matter for the field to decide for itself, the Federal Government should encourage movement toward the goal of greater professionalism.

---

[143]Bryan Kocher, "President's Letter," *Communications of the ACM,* Vol. 32(6), June 1989, p. 662.

[144]*Maintaining the Integrity of Scientific Research,* Hearings before the Subcommittee on Investigations and Oversight, Committee on Science, Space, and Technology, U.S. House of Representatives; 101st Congress, First Session, June 28, 1989 (Washington: Government Printing Office, in press).

[145]Susan B. Garland, *et al.,* "Beltway Brain Drain: Why Civil Servants Are Making Tracks," *Business Week,* January 23, 1989, p. 61.

[146]Paul A. Volcker, *et al., Leadership for America: Rebuilding the Public Service,* Report of the National Commission on the Public Service (Washington: The Commission, 1989).

V. CONCLUSION

If architects built buildings the way programmers build programs, then the first woodpecker to appear would destroy civilization.

-- ANONYMOUS[147]

No single action will magically solve the software woes afflicting the Government. Rather, the solution will be found more in a sustained and difficult series of apparently mundane -- but eventually far-reaching -- changes in policies.

One root problem is that the Government finally decides what a system should include long after the system design stage has passed. Yet this is where most of the major problems occur and, at the same time, where they can be solved most easily and cheaply. This report has focused on the malign influences of the budget process on Government procurement, since the search for funding goes far to explain why programs are pushed into development before they are ready.

What may finally force a redefinition of this procurement system is continued failure to balance the budget. The strange policy where the Government pays twice for a system -- once to buy it and again to make it work the way it was expected -- cannot be sustained in an era of multibillion-dollar shortfalls in the Treasury.

Along with the issue of better software development is the question of software regulation. The best agencies get involved with their industrial counterparts at the system design stage[148] while some agencies have yet to evaluate the impact software will have on their responsibilities. Most of this oversight, unfortunately, remains qualitative. The lack of progress in developing methods for evaluating software inhibits the full confidence that would come with an analysis based on mathematics. Development of such capability will be important in allowing agency staff to fulfill their regulatory responsibilities in the area of software.

Regulatory agencies face not only today's problems, but tomorrow's. Computer architectures today rely on a single processor to manage execution of the various application programs loaded into the system. The search for increased capability is leading to machines with multiple processors and software capable of managing simultaneous computation in thousands of processors. Further, artificial intelligence research may succeed in producing workable "adaptive" software, or a system that can learn from experience. Where regulators have to date dealt with software where changes produce similar results across all systems, adaptive intelligence means that there will be no certainty what the software will do when it is used. A situation where each separate software installation represents a single unique case may lead to regulatory overload. Adaptive intelligence in safety-critical systems may be beyond the capability of regulatory agencies to certify.

---

[147]Kocher, *op. cit.*, p. 660.

[148]Boeing Commercial Aircraft Corporation meets with FAA representatives to discuss preliminary system designs and a system hazard analysis. Discussion centers on the areas of concern identified by Boeing or the FAA. Investigators from the FAA also participate in engineering flight tests which validate solutions to design uncertainties for the Boeing engineering staff. Interview with Sam Lachmann, Requirements and Compliance Division, Boeing Commercial Aircraft Company, Renton, Washington, November 11, 1987.

Nothing recommended in this report will have any effect on the software problem in the absence of people with the capability to understand the rationale behind them and implement them. Early tremors are being felt that suggest these are the people the Government is most likely to lose in the near future. Should that occur, the problems now will seem minor by comparison.

Education needs closer attention. University curricula that equate programming with software engineering or that focus solely on hardware design are not enough to develop the specialists needed to manage large and complex systems. Ideas exist in abundance but need resources for experimentation to find out which ones actually work. Both Government and industry benefit by investing in the "human capital" that is at the center of progress in any field. Expanding the pool of talented experts should be a priority.

VI. RECOMMENDATIONS

The following are recommendations for action by the Committee on Science, Space, and Technology:

1) A Working Group on Software Development Improvement should be formed under the auspices of the Federal Coordinating Council for Science, Engineering and Technology in the Office of Science and Technology Policy.

The Working Group would focus agency software capabilities on solving software problems in the Federal Government. Initially, the Working Group should inventory available sources of expertise in Government. Cooperative ties to industry and academia should also begin. Ultimately, the Working Group should be able to provide a cross-cutting analysis of software-related resources in the Federal budget and recommendations for their most effective use in supporting improvements in software development, regulation and the education of software specialists.

The Working Group should include a chairman from the Office of Science and Technology Policy. Other members should be drawn from agencies with demonstrated software experience such as the Department of Defense, NASA, the FAA, the National Security Agency, the Internal Revenue Service, the National Science Foundation, the National Institute of Standards and Technology, the Department of Health and Human Services and the Social Security Administration. Organizationally, the Working Group may benefit if it is constituted as a new subcommittee of the present Committee on Computer Research and Applications. However, the broader responsibilities contemplated for the Working Group may require that it function separately.

The Working Group should draw on agency, industry and academic contributions to design a research program targeted on identified weaknesses in Government software, following the example of the old National Advisory Committee on Aeronautics. Absent demonstrated expertise in other agencies, technical support for the Working Group could be drawn from the National Computer Systems Laboratory of the National Institute of Standards and Technology and the Software Engineering Program of NSF's Division of Computer and Computation Research.

2) NASA should assist the Office of Federal Procurement Policy in developing and implementing a software-oriented procurement policy.

The Office of Federal Procurement Policy is working to develop the streamlined procurement regulations recommended by the Packard Commission. Whether this new regulation adopts a "system-first" perspective is not certain. NASA should be the lead agency because it has experience in the management of software-intensive systems and could expect great benefits from the success of this project.

Implementation of this policy will create important changes in an agency's operations. Congress will be faced with higher initial program costs. Progress will *appear* to occur more slowly. These two points should be clearly spelled out. The benefits will be seen in cost estimates for system production and maintenance that prove to be more accurate over the life cycle. These are more long-term and will be identified only in retrospect. For this reason, the Committee may wish to consider a pilot program on a small system procurement before revising the agency's present system.

For systems relating to an agency's administrative functions, however, purchasing a commercial product should be encouraged. In the case where the product does not precisely match an agency's requirements, it may be simpler to reorganize the agency to fit the product (humans being infinitely more adaptable than computers) rather than trying to automate the agency's present practice. This assumes, of course, that all Congressionally-mandated information is collected by the resulting organization.

3) The National Computer Systems Laboratory of the National Institute of Standards and Technology should develop the capability to analyze and evaluate proposed methods for improving Government software development before they are introduced.

A growing literature argues that the same methods used by Deming to improve the quality of hardware in a production process can be applied with equal success to software. At present, however, a debate is still occurring regarding the validity of these methods. The Laboratory is the logical point to refine these techniques and tailor them to support Government program managers and regulators.

Work is currently underway at the Laboratory's Systems and Software Technology Division that relates to improving the software life cycle. In coordination with recommendations from the Working Group on Software Development Improvement, the program should expand efforts to assist with elaboration of quantitative models for Government use when developing or regulating software and related systems. It is anticipated that the Laboratory will organize cooperative ventures with industry and academia. Consideration should be given to adopting the proposed center of excellence defined by the Triservice Software Systems Safety Working Group.

The Committee assigned responsibility for assisting business with the development of standards for computer security to the Laboratory. It should be given equivalent responsibility for adapting the

military standard for software safety[149] to commercial needs. This emerging field needs a Federal focus and the Laboratory seems ideally positioned to assume leadership in the field.

Laboratory traditions for close cooperation with industry and the provision of technical assistance to other Federal agencies closely match the skills required to effect improvements in Government software. The Committee should support efforts by the Laboratory to establish itself as a center for excellence in applied research on the software lifecycle. Alternatively, the Working Group can analyze an efficient division of these responsibilities between the Laboratory and the Centers for Excellence established by the National Science Foundation at Purdue University and the University of Florida.

4) The National Science Foundation should remain focused on basic research in next generation software problems and the improvement of education in software engineering.

The mission of the National Science Foundation is cutting-edge basic research and the support for improved education in science and technology. This recommendation attempts to play to these strengths. The Division of Computer and Computation Research is already supporting research in massively parallel processing and organizing common terms of reference necessary for "expert system" intercommunication. Both fields are important to near-term progress in next generation computing.[150] Dr. William Wulf testified before the Senate on July 26, 1989, that

> ...[A]t one time, I believed that the problem lay in getting industry to use the research results already available.... But, I am now convinced that it is the lesser of the problems. The much larger ones are that the fundamental intellectual foundation, even the appropriate mathematics, does not exist.[151]

He recommended that NSF embark on a research program in mathematics to address the gap he now perceives.

Given the plethora of recommendations for improving software engineering education already extant, NSF can serve by identifying the most promising paths to follow, and finding means for testing their contributions. Cross-fertilization between the education program at the Software Engineering Institute and NSF should be pursued. NSF should also contribute appropriately to curriculum projects now underway with professional societies in the field. NSF should be directed to press for stronger consideration of ethical norms in new curricula.

Two points are highlighted for special attention. NSF should assure that computer science or software engineering curricula should expose students to systems engineering concepts throughout their education. Also, NSF should investigate the suitability of organizing software engineering programs, separate from but affiliated with the computer science department, at some universities.

---

[149]MIL-STD-882B, Notice 1.

[150]Interview with Dr. Peter Freeman, Director, Division of Computer and Computation Research, National Science Foundation, December 12, 1988.

[151]William Wulf, Letter to staff, page 3.

5) Activities at the Software Engineering Institute should be supported by a broader spectrum of Federal agencies, or similar organizations should be developed by non-DOD agencies.

The Institute was established by the Department of Defense to serve as the technology transfer agent for advances in the Consolidated Software Initiative. Discussions with SEI staff, however, about their activities were not limited to military needs. Applying the SEI methodology for evaluating the software capability of DOD contractors, for example, is just as useful to an agency like FDA.

Most of the major changes in software engineering methods appear to have universal application, and the SEI is specifically organized to find the most effective means for disseminating better practice. The strong identification with DOD may inhibit the willingness of others to take advantage of SEI.

Though the institute supports all Government agencies, providing support may give SEI resources sufficient to expand assistance to medical device manufacturers or nuclear power plant operators seeking software assistance. Though SEI is already in being, and has to date performed well, FDA should evaluate developing a similar capability focused on the specialized need of the small medical device manufacturer for software quality assurance.

VII. ACKNOWLEDGMENTS

The Subcommittee staff wishes to acknowledge the assistance of the following people in the field of computer software for their assistance in the preparation of this report. The perspectives and information they offered were invaluable to the completion of this study.

The staff remains responsible for errors of fact or omission contained in the study and for recommendations or statements of policy.

Dr. Harold Podell	Committee on Science, Space, and Technology
David Clement	Committee on Science, Space, and Technology
Ms. Delores Wallace	National Institute of Standards and Technology
Mr. Roger Martin	National Institute of Standards and Technology
Dr. Peter Freeman	National Science Foundation
Dr. William Wulf	National Science Foundation
Mr. William Wilson	National Aeronautics and Space Administration
Mr. Donald Sova	National Aeronautics and Space Administration
Mr. Dennis Horn	National Aeronautics and Space Administration
Mr. Frank Buzzard	National Aeronautics and Space Administration

Mr. Al Habbestad	Federal Aviation Administration
Mr. Ken Meyer	Federal Aviation Administration
Mr. James Treacy	Federal Aviation Administration
Mr. Michael DeWalt	Federal Aviation Administration
Mr. Philip Gill	Federal Aviation Administration
Mr. Russ Spadea	Federal Aviation Administration
Mr. Tom Grygotis	Federal Aviation Administration
Mr. Roger Schneider	Food and Drug Administration
Mr. M. Frank Houston	Food and Drug Administration
Dr. Judith Prewitt	Institute for Electrical and Electronic Engineers
Mr. Nick Stewart	American Society for Quality Control
Mr. Alexander A.J. Hoffman	Association for Computing Machinery
Mr. W. Richards Adrion	Association for Computing Machinery
Dr. Peter Neumann	SRI International
Mr. John Musa	AT&T Bell Laboratories
Mr. Jerry Smith	National Security Industrial Association
Mr. Michael Brown	Naval Surface Warfare Center
Col. Joseph Greene	U.S. Air Force
Mr. Sam Lachmann	Boeing Commercial Aircraft Company
Mr. Gary Chenkovich	Boeing Commercial Aircraft Company
Mr. Taz Daughtrey	Babcock & Wilcox
Ms. Charlotte Guthrie	Power Computing, Inc.
Mr. Roger Fujii	Logicon, Inc.
Dr. Gerald McDonald	BDM Corporation
Ms. Rita Soule	Martin Marietta Corporation
Mr. John Summers	MITRE Corporation
Mr. Gene Loeffler	RMS Technologies
Mr. Fletcher Buckley	GE Aerospace
Dr. David Weiss	Software Productivity Consortium
Mr. Watts Humphrey	Software Engineering Institute
Mr. Norman Gibbs	Software Engineering Institute
Mr. James Withey	Software Engineering Institute

Mr. Bill Wood	Software Engineering Institute
Mr. Clyde Chittister	Software Engineering Institute
Mr. Larry Gresham	Software Engineering Institute
Dr. Harlan Mills	Information Systems Institute
Dr. Nancy Leveson	University of California at Irvine
Dr. Jesse Poore	University of Tennessee
Prof. Jonathan Jacky	University of Washington
Prof. Donald Gotterbarn	The Wichita State University
Mr. J.J. BloomBecker	National Center for Computer Crime Data